

\*\* additional items to be added to site map

# All Components

* Identify all the data: states and props
* Identify the flow of data
* Identify all functions and handlers
* Identify all components
* Modify all components to make smaller components
* Add names to component elements that make sense for form input
* Add client-side and server-side validation
* Modify all components to make functional components and make parent state component
* Add lifecycle methods to all classes
* Analyze all lifecycle methods
* Add tests to all functions
* Send Email for actions needed for Donations & Pickups

Let’s transform all these tasks to user-value statements in Trello cards.

Try to reword the tasks in the following format:

As a…

I want…

So that…

For example

As a donor

I want to save profile information

So that I don’t have to repeat it each time I publish an item

# All Components

* Identify all the data: states and props
* Identify the flow of data
* Identify all functions and handlers
* Identify all components
* Modify all components to make smaller components
* Add names to component elements that make sense for form input
* Add client-side and server-side validation
* Modify all components to make functional components and make parent state component
* Add lifecycle methods to all classes
* Analyze all lifecycle methods
* Add tests to all functions

# Navbar

## Mobile Responsive

* ~~Add icon for toggle menu button~~
* ~~Move Donate to top, make logo smaller~~
* ~~For ipad, remove toggle, make logo smaller and possibly links smaller~~
* ~~Move keyword search and city/zip search to shop when collapse~~

# Footer

* Add US copyright symbol and year
* Add logo flower

# Shop

## Pending User Actions

* ~~Show actions needed for Donations & Pickups~~

## Test

* ~~Add unit tests for search conditions~~

# Sign Up Modal

## Validate

* ~~Validate all user data~~
* ~~Implement UI messages for user data~~
* ~~Redo passport and user authentication using local and google.~~
* ~~Understand every bit of code, comment all code.~~

# Login Modal

## Validate

* ~~Validate all user data~~
* ~~Implement UI messages for user data~~
* ~~Redo passport and user authentication using local and google.~~
* ~~Understand every bit of code, comment all code.~~

# Donate Form

## File Upload

* ~~Add conditional for file upload if dev or production~~
* ~~Add streaming to display picture~~
* ~~Upload file upon click Donate~~
* ~~Add disclosure for donation retention~~
* ~~Picture and input text scanned for hate and pornography~~

# I’m interested Modal

* ~~Create Mockup~~
* ~~Implement~~
* Send Email for actions needed for Donations & Pickups

# My Donations

* ~~Create Mockup~~
* ~~Implement~~

# My Pickups

* ~~Create Mockup~~
* ~~Implement~~

# User Profile

* ~~Create Mockup~~
* ~~Implement~~

# About US

* ~~Create Mockup~~
* ~~Implement~~

## FAQ

* ~~Add how to use steps~~

# **Data Models**

## **Document**

* **data models**
* **use cases**

# **Redux**

* **Add Redux: reducers, actions and store.**
* **Use Redux to feed stateful components**

# **Testing**

## **Hardware**

* **Desktop**
* **Laptop**
* **Ipad**
* **mobile**

# Legal

* Obtain US copyright
* ~~Security disclosure~~
* ~~Validate picture for no hate or pornography~~
* ~~Validate user input for no hate or pornography~~
* ~~Register with AWS for storage space~~
* Register with Mongodb (Mlab) for storage space
* ~~Domain Name~~
* ~~Deploy Domain~~

JEST – unit test framework

Cucumber

Make a jest branch

Figure out what order of priority

Make cards with business

Estimate stories in terms of complexity in points

Adding an icon in relation to the pointer

What is Test Driven Development? Coding of features and tests go hand in hand.

1. Write a unit test
2. Run the test. See it fail.
3. Write the feature code to pass the test
4. Refactor the code

What is Behavior Driven Development (BDD)?

* A variation of TDD that tests for user scenarios
* Given, when, then
* Given notes, when deleting, then remove a note
* Bdd consists of scenarios/specifications

Jest + Enzyme = TDD with React

* Jest (Facebook) – test JavaScript, includes snapshot testing, coverage and mocking (check if something was called).
* Enzyme (Airbnb) – Allow us to work with specific components. Testing of manipulating React components and DOM behavior. Ability to find specific items in component itself.

**Installing enzyme**

Install enzyme with jest cli: npm install enzyme jest-cli@20.0.4 --save-dev

Install react bootstrap: npm install react-bootstrap –save

Run tests: npm run test (will run test with test.js extension)

If problems

remove node folder: rm -rf node\_modules

To reinstall node modules: npm i

Install react test render: npm i react-test-render --save-dev

Install adapter for enzyme and react: npm i enzyme-adapter-react-16 --save-dev

**Testing with jest**

Shallow – helper function from enzyme, will only render out JSX, no child components.

it('renders correctly', () => {

expect(app).toMatchSnapshot();

})

Important: Enzyme Adapter

Section 2, Lecture 8

Please keep this in mind!

In order to use the most current version of React > 16, we now need to install "enzyme adapters" to provide full compatibility with React.

There's two options to take. The first is quicker, the second is recommended:

1. You can use the same versions for the node\_modules libraries in the package.json files of the original repo:

Or you can:

2. Install the adapter for enzyme and react:

npm i enzyme-adapter-react-16 --save-dev

Next, add a src/tempPolyfills.js file to create the global request animation frame function that React now depends on.

src/tempPolyfills.js should contain the following contents:

const requestAnimationFrame = global.requestAnimationFrame = callback => {

setTimeout(callback, 0);

}

export default requestAnimationFrame;

Finally, add a src/setupTests.js file to configure the enzmye adapter for our tests. The disableLifecyleMethods portion is needed to allow us to modify props through different tests.

src/setupTests.js should contain the following contents:

import requestAnimationFrame from './tempPolyfills';

import { configure } from 'enzyme';

import Adapter from 'enzyme-adapter-react-16';

configure({ adapter: new Adapter(), disableLifecycleMethods: true });

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Let's keep coding!

// check if component renders correctly

// to create a jest unit test, we use jest method 'it' function

// it takes two args, test desc and test

it('renders correctly', () => {

// expect jest method, allows us to check if parts of this app component meet certain conditions

// testing conditions are asserted by chaining other jest methods after the expect method

// toMatchSnapshot is a key method of snapshots. Snapshots keep a recorded history of your react components,

// everytime a change is made in the application snapshot will update and make sure previous components

// in testing files match your new one

// using snapshot we can test the rendered JSX as a whole

expect(app).toMatchSnapshot();

});

// once me make changes to the rendered component of App.js, the test will fail,

// because the rendered component is different than the snapshot.

// We can then confirm the change and by pressing u to update the snapshot

it('initializes the `state` with an empty list of gifts', () => {

expect(app.state().gifts).toEqual([]);

});

it('adds a new gift to `state` when clicking the `add gift` button', () => {

// find add gift button and click

// find method lets us find child or inner components by JSX tags or by className

app.find('.btn-add').simulate('click');

expect(app.state().gifts).toEqual([{id: 1}]);

});

it('adds a new gift to the rendered list when clicking the `add gift` button', () => {

app.find('.btn-add').simulate('click');

expect(app.find('.gift-list').children().length).toEqual(2);

})

* With BDD it’s a common practice to group a bunch of tests together.
* Use Jest Global **describe** method to group tests together,
* \*\*\* Snapshot becomes obsolete
* Press u to update
* Press a to rerun tests